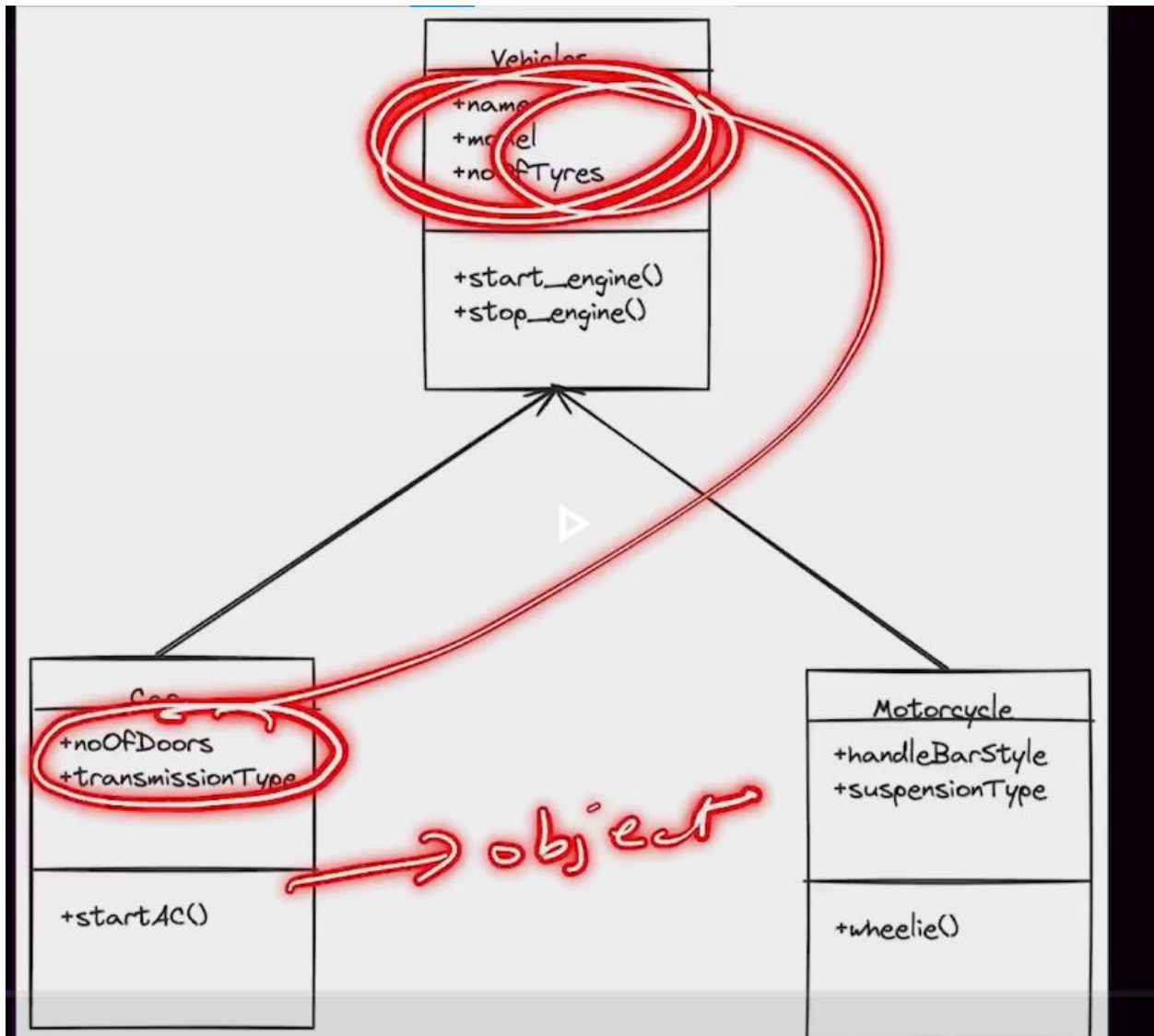


## 6. Implementing Inheritance in C++



	Derived Class	Derived Class	Derived Class
Base Class	Private Mode	Protected Mode	Public Mode
Private	Not Inherited	Not Inherited	Not Inherited
Protected	Private	Protected	Protected
Public	Private	Protected	Public

```

4   using namespace std;
5
6   class Vehicle
7   {
8   public:
9       string name;
10      string model;
11      int noOfTyres;
12
13      Vehicle(string _name, string _model, int _noOfTyres)
14      {
15          cout << "I am inside Vehicle ctor" << endl;
16          this->name = _name;
17          this->model = _model;
18          this->noOfTyres = _noOfTyres;
19      }
20
21  public:
22      void start_engine()
23      {
24          cout << "Engine is starting " << name << " " << model << endl;
25      }
26
27      void stop_engine()
28      {
29          cout << "Engine is stopping " << name << " " << model << endl;
30      }
31  };
32

```

```

33  class Car : public Vehicle
34  {
35  public:
36      int noOfDoors;
37      string transmissionType;
38
39      Car(string _name, string _model, int _noOfTyres, int _noOfDoors, string _transmissionType) : Vehicle(_name, _model, _noOfTyres)
40      {
41          cout << "I am inside Car ctor" << endl;
42          this->noOfDoors = _noOfDoors;
43          this->transmissionType = _transmissionType;
44      }
45
46      void startAC()
47      {
48          cout << "AC has started of " << name << endl;
49      }
50  };
51
52  int main()
53  {
54      Car A("Maruti 800", "LXI", 4, 4, "Manual");
55      return 0;
56  }

```

Output:

```

(base) lakshaykumar@Lakshays-MacBook-Air output % ./"VehicleInheritance"
I am inside Vehicle ctor
I am inside Car ctor
(base) lakshaykumar@Lakshays-MacBook-Air output %

```

Child class banane se pahle base class ka constructor call hota hai

----

```

52  int main()
53  {
54      Car A("Maruti 800", "LXI", 4, 4, "Manual");
55      A.start_engine();
56      A.startAC();
57      A.stop_engine();
58      return 0;
59  }

```

TERMINAL   PORTS   PROBLEMS   OUTPUT   DEBUG CONSOLE

```

I am inside Car ctor
• (base) lakshaykumar@Lakshays-MacBook-Air output % cd "/Users/lakshaykumar/Desktop"
./"VehicleInheritance"
• (base) lakshaykumar@Lakshays-MacBook-Air output % ./"VehicleInheritance"
I am inside Vehicle ctor
I am inside Car ctor
Engine is starting Maruti 800 LXI
AC has started of Maruti 800
Engine is stopping Maruti 800 LXI
○ (base) lakshaykumar@Lakshays-MacBook-Air output %

```

---

Base class ke private member ko inheriate kar rahe public child class mai toh ,member ko access nahi kar sakte child class ,agar private member ko access karna child mai toh GETTER () AUR SETTER () METHOD Ko use Kareng

```

G: VehicleInheritance.cpp > Car
1  #include <iostream>
2  #include <string>
3
4  using namespace std;
5
6  class Vehicle
7  {
8  private:
9      string name;
10
11 public:
12     string model;
13     int noOfTyres;
14
15     string getName()
16     {
17         return this->name;
18     }
19
20     Vehicle(string _name, string _model, int _noOfTyres)
21     {
22         cout << "I am inside Vehicle ctor" << endl;
23         this->name = _name;
24         this->model = _model;
25         this->noOfTyres = _noOfTyres;
26     }
27
28 public:
29     void start_engine()
30     {

```

```

42 public:
43     int noOfDoors;
44     string transmissionType;
45
46     Car(string _name, string _model, int _noOfTyres, int _noOfDoors, string _transmissionType) : Vehicle(_name, _model, _noOfTyres)
47     {
48         cout << "I am inside Car ctor" << endl;
49         this->noOfDoors = _noOfDoors;
50         this->transmissionType = _transmissionType;
51     }
52
53     void startAC()
54     {
55         cout << "AC has started of " << getName() << endl;
56     }
57 };
58

```

→ Private member of a class  
 on In or for getter() and setter() method  
 on use on In

```

TERMINAL  PORTS  PROBLEMS  OUTPUT  DEBUG CONSOLE
Engine is stopping Maruti 800 LXI
(base) lakshaykumar@Lakshays-MacBook-Air output % cd "/Users/lakshaykumar/Desktop/codehelp/Supra-LLD/C++ Codes/output"
./"VehicleInheritance"
I am inside Vehicle ctor
I am inside Car ctor
Engine is starting Maruti 800 LXI
AC has started of Maruti 800
Engine is stopping Maruti 800 LXI
(base) lakshaykumar@Lakshays-MacBook-Air output %

```

Compiled successfully!

Base class ke Protected member ko child public mode class mai access karenge toh ,child class mai as protected member aa jayeag.

```
VehicleInheritance.cpp > Car > startAC()
1  #include <iostream>
2  #include <string>
3
4  using namespace std;
5
6  class Vehicle
7  {
8  protected:
9      string name;
10
11  public:
12      string model;
13      int noOfTyres;
14      Vehicle(string _name, string _model, int _noOfTyres)
15      {
16          cout << "I am inside Vehicle ctor" << endl;
17          this->name = _name;
18          this->model = _model;
19          this->noOfTyres = _noOfTyres;
20      }
21
22  public:
23      void start_engine()
24      {
25          cout << "Engine is starting " << name << " " << model << endl;
26      }
27
28      void stop_engine()
29      {
30          cout << "Engine is stopping " << name << " " << model << endl;
31      }
32  };
33
34  class Car : public Vehicle
35  {
36  public:
37      int noOfDoors;
38      string transmissionType;
39
40      Car(string _name, string _model, int _noOfTyres, int _noOfDoors, string _transmissionType) : Vehicle(_name, _model, _noOfTyres)
41      {
42          cout << "I am inside Car ctor" << endl;
43          this->noOfDoors = _noOfDoors;
44          this->transmissionType = _transmissionType;
45      }
46
47      void startAC()
48      {
49          cout << "AC has started of " << name << endl;
50      }
51  };
52
53  int main()
54  {
55      Car A("Maruti 800", "LXI", 4, 4, "Manual");
56      A.start_engine();
57      A.startAC();
58      A.stop_engine();
59      return 0;
60  }
```



-----

Parent class se protected member child class mai inheriate toh ho jayega as protected member lekin private ke jais behave karega. Child class mai acces hoga lekin jab child ka object bana kar ya base class ka object bana ke karenge toh error aaega

```
VehicleInheritance.cpp > Vehicle
46
47     void startAC()
48     {
49         cout << "AC has started of " << name << endl;
50     }
51 };
52
53 int main()
54 {
55     Car A("Maruti 800", "LXI", 4, 4, "Manual");
56     A.start_engine();
57     A.startAC();
58     A.stop_engine();
59
60     Vehicle v(kjenfkef);
61     v.name;
62     return 0;
63 }
```

```
VehicleInheritance.cpp > Car
33
34 class Car : public Vehicle
35 {
36 public:
37     int noOfDoors;
38     string transmissionType;
39
40     Car(string _name, string _model, int _noOfTyres, int _noOfDoors, string _transmissionType) : Vehicle(_name, _model, _noOfTyres)
41     {
42         cout << "I am inside Car ctor" << endl;
43         this->noOfDoors = _noOfDoors;
44         this->transmissionType = _transmissionType;
45     }
46
47     void startAC()
48     {
49         cout << "AC has started of " << name << endl;
50     }
51 };
52
53 int main()
54 {
55     Car A("Maruti 800", "LXI", 4, 4, "Manual");
56     A.start_engine();
57     A.startAC();
58     A.stop_engine();
59     cout << A.name;
60     return 0;
61 }
```

→ error



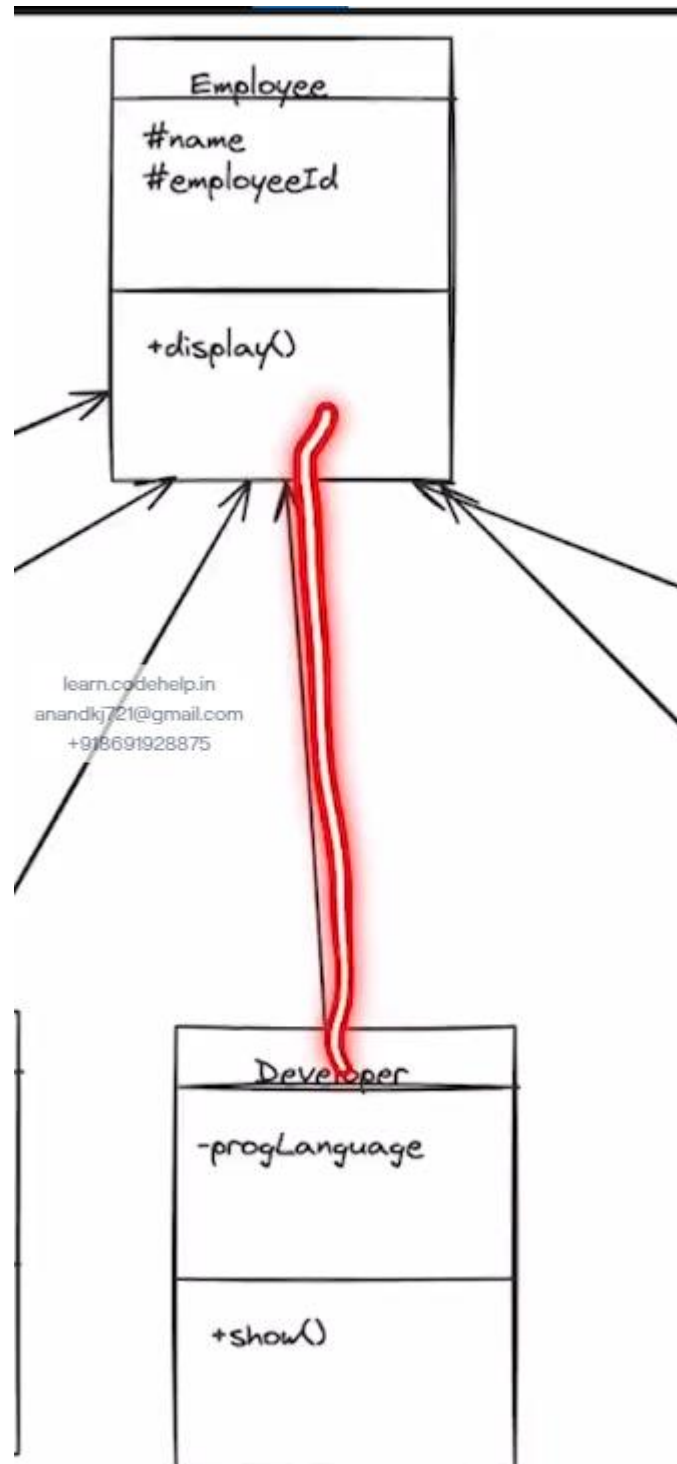
Protected member parent se child derived class mai aa jayega lekin main class access nahi hoga.

---

Base class ka constructor derived class ke constructor se pahle call hoga .lekin jab hum destructor call opposite hota hai,pahle derived class ka destructor call hoga uske badd mai Base class ka destructor

---

## Single Inheritance





```

1  #include <iostream>
2  #include <string>
3
4  // Base class for Single Inheritance
5  class Employee
6  {
7  protected:
8      std::string name;
9      int employeeId;
10
11 public:
12     Employee(const std::string &empName, int empId) : name(empName), employeeId(empId)
13     {
14     }
15
16     void display() const
17     {
18         std::cout << "Employee: " << name << ", ID: " << employeeId << std::endl;
19     }
20 };
21
22 // Derived class for Single Inheritance
23 class Developer : public Employee
24 {
25 private:
26     std::string programmingLanguage;
27
28 public:
29     Developer(const std::string &empName, int empId, const std::string &lang)
30         : Employee(empName, empId), programmingLanguage(lang) {
31     }
32
33     void show() const
34     {
35         display();
36         std::cout << "Specialization: Developer, Programming Language: " << programmingLanguage << std::endl;
37     }
38 };
39

```

```

172 int main()
173 {
174     // Single Inheritance
175     Developer dev("Ramu Kaka", 101, "C++");
176     dev.show();
177

```

## Output

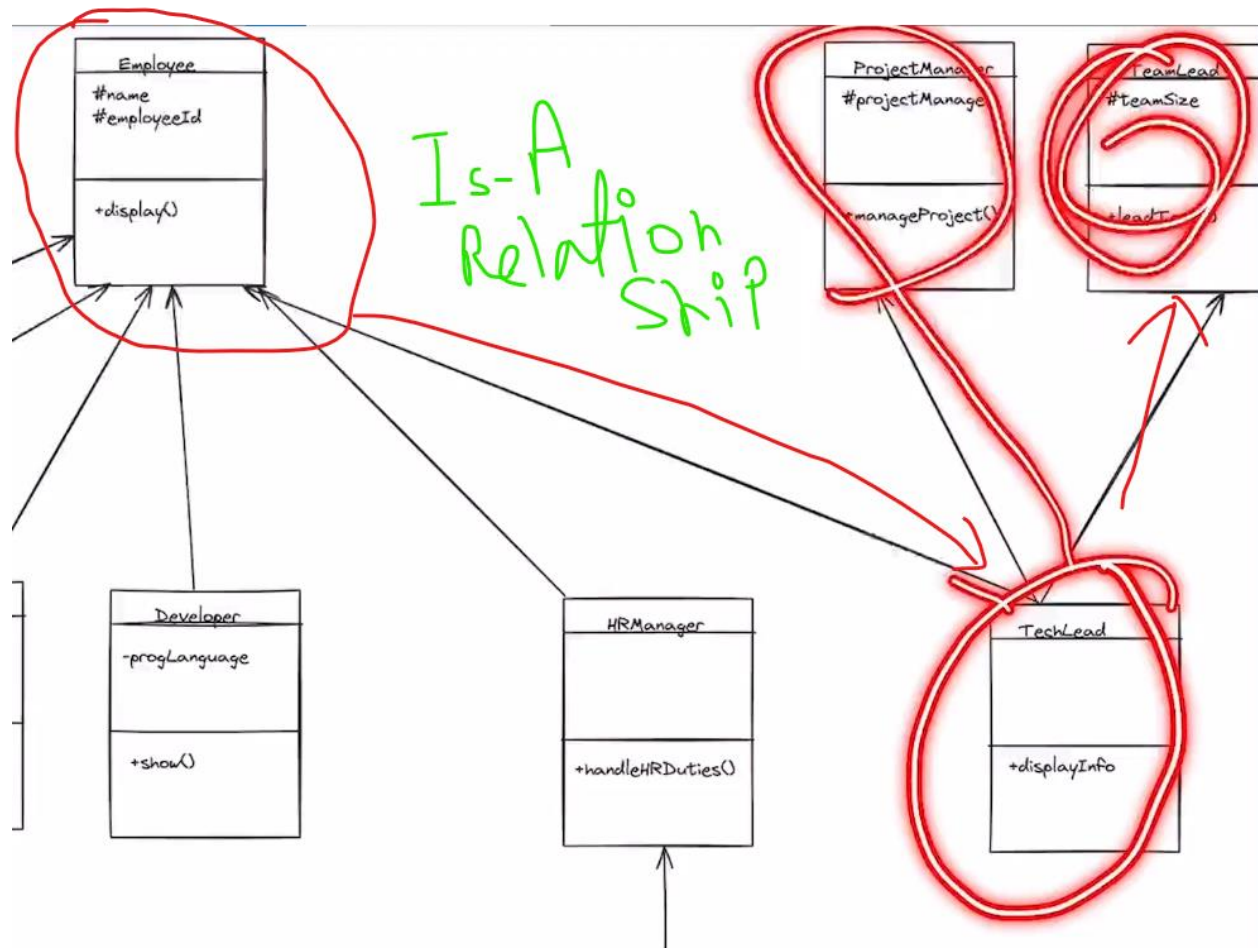
```

• (base) lakshaykumar@Lakshays-MacBook-Air output % ./"VehicleInheritance"
I am inside Vehicle ctor
I am inside Car ctor
Engine is starting Maruti 800 LXI
AC has started of Maruti 800
Engine is stopping Maruti 800 LXI
I am inside Car dtor
I am inside Vehicle dtor
• (base) lakshaykumar@Lakshays-MacBook-Air output % cd "/Users/lakshaykumar/Desktop/codehelp/Supra-LLD/C++ Codes/output"
./"TypesOfInheritance"
• (base) lakshaykumar@Lakshays-MacBook-Air output % ./"TypesOfInheritance"
Employee: Ramu Kaka, ID: 101
Specialization: Developer, Programming Language: C++
• (base) lakshaykumar@Lakshays-MacBook-Air output %

```

---

## Multiple Inheritance



```
5  class Employee
6  {
7  protected:
8      std::string name;
9      int employeeId;
10
11 public:
12     Employee(const std::string &empName, int empId) : name(empName), employeeId(empId)
13     {
14         // std::cout << __FUNCTION__ << std::endl;
15     }
16
17     void display() const
18     {
19         std::cout << "Employee: " << name << ", ID: " << employeeId << std::endl;
20     }
21 };
22
```

```

41
42 // Base classes for Multiple Inheritance
43 class ProjectManager
44 {
45 protected:
46     std::string projectManaged;
47
48 public:
49     ProjectManager(const std::string &project) : projectManaged(project) {}
50
51     void manageProject() const
52     {
53         std::cout << "Project Manager managing project: " << projectManaged << std::endl;
54     }
55 };
56

```

```

57 class TeamLead
58 {
59 protected:
60     int teamSize;
61
62 public:
63     TeamLead(int size) : teamSize(size) {}
64
65     void leadTeam() const
66     {
67         std::cout << "Team Lead leading a team of " << teamSize << " members." << std::endl;
68     }
69 };

```

```

70
71 // Derived class for Multiple Inheritance
72 class TechLead : public Employee, public ProjectManager, public TeamLead
73 {
74 public:
75     TechLead(const std::string &empName, int empId, const std::string &project, int teamSize)
76         : Employee(empName, empId), ProjectManager(project), TeamLead(teamSize) {}
77
78     void displayInfo() const
79     {
80         display();
81         manageProject();
82         leadTeam();
83     }
84 };
85

```

```

172 int main()
173 {
174     // Single Inheritance
175     // Developer dev("Ramu Kaka", 101, "C++");
176     // dev.show();
177
178     // // Multiple Inheritance
179     TechLead techLead("Anna Dev", 202, "Project X", 5);
180     techLead.displayInfo();
181

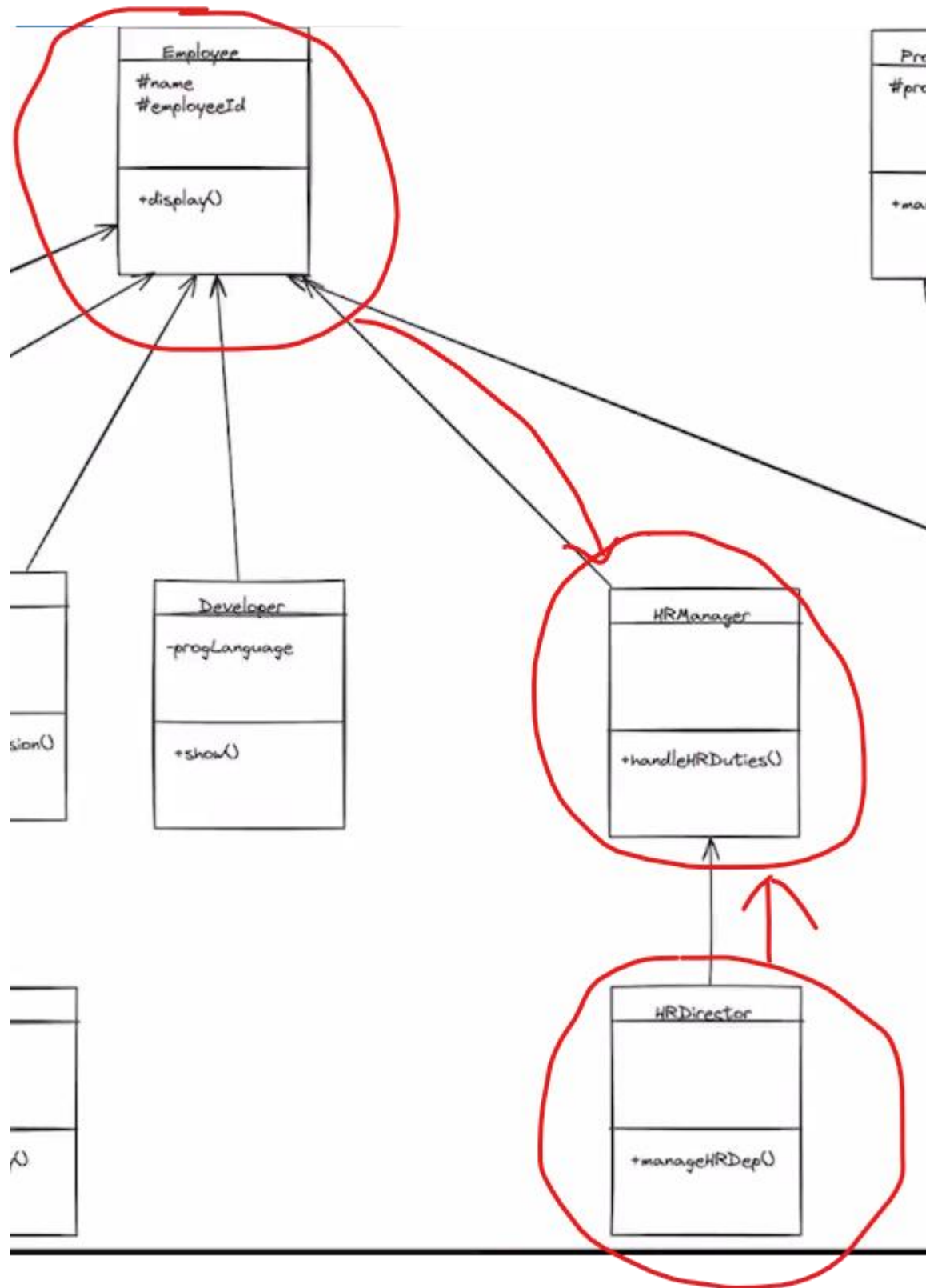
```

## Output

```
TERMINAL  PORTS  PROBLEMS  OUTPUT  DEBUG CONSOLE

• (base) lakshaykumar@Lakshays-MacBook-Air output % cd "/Users/lakshaykumar/Desktop/codehelp/Supra-LLD/C++ Codes/output"
./"TypesOfInheritance"
• (base) lakshaykumar@Lakshays-MacBook-Air output % ./"TypesOfInheritance"
Employee: Anna Dev, ID: 202
Project Manager managing project: Project X
Team Lead leading a team of 5 members.
○ (base) lakshaykumar@Lakshays-MacBook-Air output %
```

## Multilevel inheritance



```

86 // Base class for Multi-level Inheritance
87 class HRManager : public Employee
88 {
89 public:
90     HRManager(const std::string &empName, int empId) : Employee(empName, empId) {}
91
92     void handleHRDuties() const
93     {
94         std::cout << "HR Manager handling human resources duties." << std::endl;
95     }
96 };
97
98 // Derived class for Multi-level Inheritance
99 class HRDirector : public HRManager
100 {
101 public:
102     HRDirector(const std::string &empName, int empId) : HRManager(empName, empId) {}
103
104     void manageHRDepartment() const
105     {
106         std::cout << "HR Director managing the HR department." << std::endl;
107     }
108 };

```

```

179 int main()
180 {
181     // Single Inheritance
182     // Developer dev("Ramu Kaka", 101, "C++");
183     // dev.show();
184
185     // // Multiple Inheritance
186     // TechLead techLead("Anna Dev", 202, "Project X", 5);
187     // techLead.displayInfo();
188
189     // // Multi-level Inheritance
190     HRDirector hrDirector("Lucy Madam", 303);
191     hrDirector.handleHRDuties();
192     hrDirector.manageHRDepartment();
193 }

```



```
TERMINAL  PORTS  PROBLEMS  OUTPUT  DEBUG CONSOLE

• (base) lakshaykumar@Lakshays-MacBook-Air output % cd "/Users/lakshaykumar/Desktop/codehelp/Supra-LLD/C++ Codes/output"
./"TypesOfInheritance"
• (base) lakshaykumar@Lakshays-MacBook-Air output % ./"TypesOfInheritance"
Employee: Anna Dev, ID: 202
Project Manager managing project: Project X
Team Lead leading a team of 5 members.
• (base) lakshaykumar@Lakshays-MacBook-Air output % cd "/Users/lakshaykumar/Desktop/codehelp/Supra-LLD/C++ Codes/output"
./"TypesOfInheritance"
• (base) lakshaykumar@Lakshays-MacBook-Air output % ./"TypesOfInheritance"
Employee
HRManager
HRDirector
HR Manager handling human resources duties.
HR Director managing the HR department.
• (base) lakshaykumar@Lakshays-MacBook-Air output %
```

```
88 class HRManager : public Employee
89 {
90 public:
91     HRManager(const std::string &empName, int empId) : Employee(empName, empId)
92     {
93         std::cout << __FUNCTION__ << std::endl;
94     }
95
96     void handleHRDuties() const
97     {
98         std::cout << "HR Manager handling human resources duties." << std::endl;
99     }
100 };
101
102 // Derived class for Multi-level Inheritance
103 class HRDirector : public HRManager
104 {
105 public:
106     HRDirector(const std::string &empName, int empId) : HRManager(empName, empId)
107     {
108         std::cout << __FUNCTION__ << std::endl;
109     }
110
111     void manageHRDepartment() const
112     {
113         std::cout << "HR Director managing the HR department." << std::endl;
114     }
115 };
```

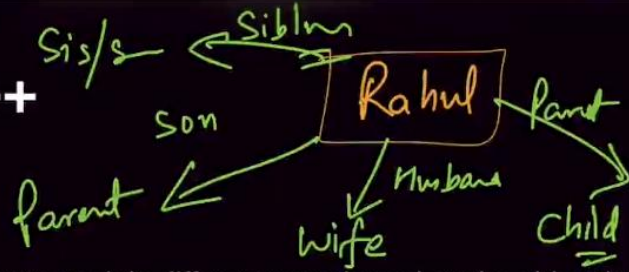
→ Class on Name Print  
ET GUNOII



## 7. Polymorphism in C++

### Polymorphism in C++

1. Similar to Polymorphism in Life.
2. Polymorphism = Many Forms.
3. The ability of a single function or Operator to work in different ways based on the object it is acting upon or actual need.
4. A phenomenon that allows an object to have several different forms and behaviours.
5. Types
  1. Compile Time Polymorphism. *Static*
  2. Runtime Polymorphism. *Dynamic*



### Static Polymorphism

1. Aka, Compile Time Polymorphism.
2. Types
  1. Function Overloading
  2. Operator Overloading

# Function Overloading

1. Overloading occurs when a class contains multiple methods sharing a name but differing in argument count or argument type.

```
class Add {  
public:  
    int sum(int x, int y) { return x + y; }  
    double sum(double x, double y) { return x + y; }  
};
```

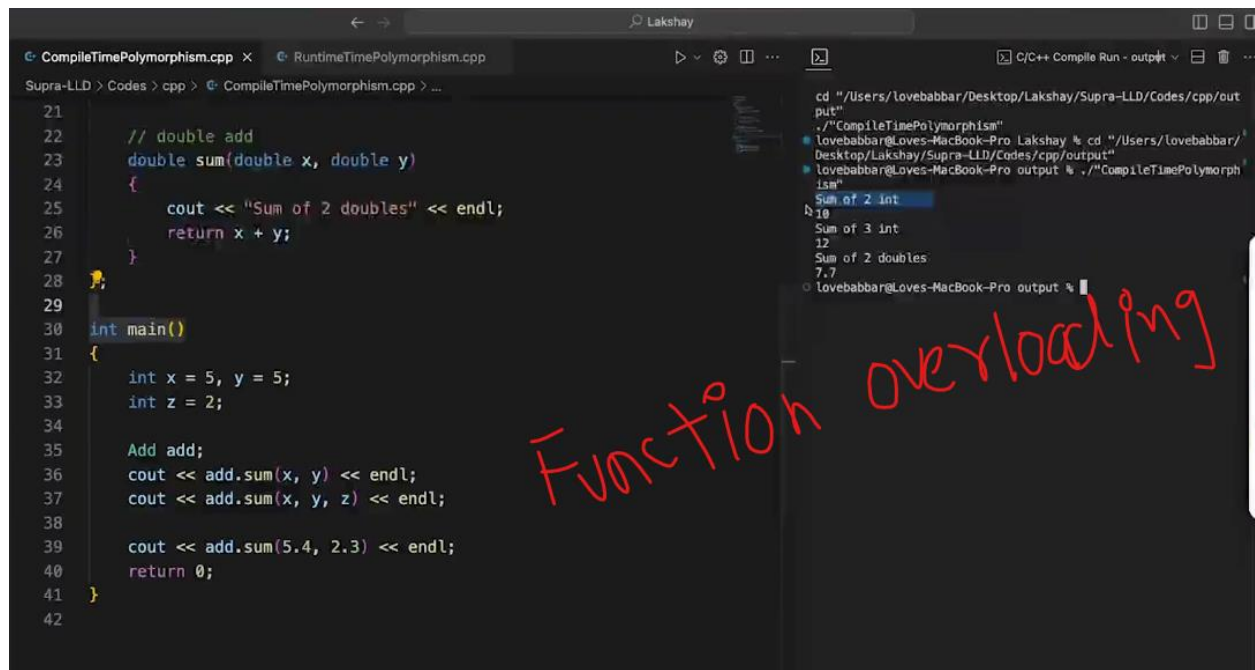
int sum(int x, int y, int z)

→

Int ⊕

double  
+

```
5  class Add  
6  {  
7  public:  
8      // x, y, two int addition  
9      int sum(int x, int y)  
10     {  
11         return x + y;  
12     }  
13  
14     // x, y, z, three int add  
15     int sum(int x, int y, int z)  
16     {  
17         return x + y + z;  
18     }  
19  
20     // double add  
21     double sum(double x, double y)  
22     {  
23         return x + y;  
24     }  
25 };
```



The screenshot shows a C++ IDE with two tabs: 'CompileTimePolymorphism.cpp' and 'RuntimeTimePolymorphism.cpp'. The 'CompileTimePolymorphism.cpp' tab is active, displaying the following code:

```
21
22 // double add
23 double sum(double x, double y)
24 {
25     cout << "Sum of 2 doubles" << endl;
26     return x + y;
27 }
28
29
30 int main()
31 {
32     int x = 5, y = 5;
33     int z = 2;
34
35     Add add;
36     cout << add.sum(x, y) << endl;
37     cout << add.sum(x, y, z) << endl;
38
39     cout << add.sum(5.4, 2.3) << endl;
40     return 0;
41 }
42
```

The output window on the right shows the following output:

```
cd "/Users/lovebabbar/Desktop/Lakshay/Supra-LLD/Codes/cpp/output"
./"CompileTimePolymorphism"
lovebabbar@Loves-MacBook-Pro Lakshay % cd "/Users/lovebabbar/Desktop/Lakshay/Supra-LLD/Codes/cpp/output"
lovebabbar@Loves-MacBook-Pro output % ./"CompileTimePolymorphism"
Sum of 2 Int
10
Sum of 3 Int
12
Sum of 2 doubles
7.7
lovebabbar@Loves-MacBook-Pro output %
```

A large red handwritten text "Function overloading" is written across the output window.

# Operator Overloading

1. In C++, when operators are overloaded, they execute user-defined functions whenever used, allowing for customised behaviour.

```
class Complex {
public:
    int real, imag;
    Complex(int r = 0, int i = 0) {real = r; imag = i;}
    Complex operator + (const Complex &obj) {
        Complex res;
        res.real = real + obj.real;
        res.imag = imag + obj.imag;
        return res;
    }
};
```

## Operator Overloading

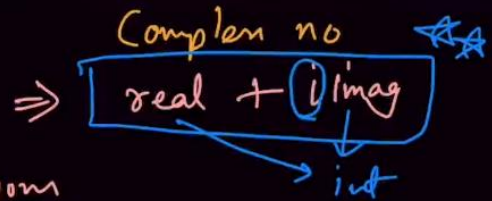
add  $\Rightarrow$  Rule

- ① real parts addition
- ② Imag " "

$$A = 2 + i5$$

$$B = 2 + i3$$

$$C = A + B \Rightarrow (2+2) + i(5+3) = 4 + i8$$



## Operator Overloading

add  $\Rightarrow$  Rule

- ① real parts addition
- ② Imag " "

$$A = 2 + i5$$

$$B = 3 + i3$$

$$C = A + B$$

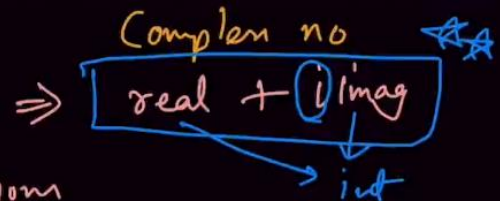
$$A \oplus B$$

fun (Complen B)

< this  $\rightarrow$  real + B.real  
↓

>

A.fun(B)



```

30 class Complex
31 {
32 public:
33     int real;
34     int imag;
35
36     Complex()
37     {
38         real = imag = -1;
39     }
40
41     Complex(int r, int i) : real(r), imag(i){};
42
43     // syntax
44     // Ret_type operator <op> (args){
45     //     // mlkdmk
46     //     return <>
47     // }
48
49     Complex operator+(const Complex &B)
50     {
51         /// this -> A instance
52         Complex temp;
53         temp.real = this->real + B.real;
54         temp.imag = this->imag + B.imag;
55         return temp;
56     }

```

```

CompileTimePolymorphism.cpp x RuntimeTimePolymorphism.cpp
Supra-LLD > Codes > cpp > CompileTimePolymorphism.cpp > main()
52     Complex temp;
53     temp.real = this->real + B.real;
54     temp.imag = this->imag + B.imag;
55     return temp;
56 }
57
58 void print()
59 {
60     printf("%d + i%d\n", this->real, this->imag);
61 }
62 };
63
64 int main()
65 {
66     Complex A(2, 5);
67     A.print();
68     Complex B(3, 3);
69     B.print();
70
71     Complex C = A + B;
72     C.print();
73 }

```

```

cd "/Users/lovebabbar/Desktop/Lakshay/Supra-LLD"
put"
./"CompileTimePolymorphism"
lovebabbar@Loves-MacBook-Pro Lakshay % cd "/Users/lovebabbar/Desktop/Lakshay/Supra-LLD/Codes/cpp/output"
lovebabbar@Loves-MacBook-Pro output % ./"CompileTimePolymorphism"
[2 + i5]
[3 + i3]
[5 + i8]
lovebabbar@Loves-MacBook-Pro output %

```

----

```

30 class Complex
31 {
32 public:
33     int real;
34     int imag;
35
36     Complex()
37     {
38         real = imag = -1;
39     }
40
41     Complex(int r, int i) : real(r), imag(i){};
42
43     // syntax
44     // Ret_type operator <op> (args){
45     //     // mlkdmk
46     //     return <>
47     // }
48
49     Complex operator+(const Complex &B)
50     {
51         /// this -> A instance
52         Complex temp;
53         temp.real = this->real + B.real;
54         temp.imag = this->imag + B.imag;
55         return temp;
56     }

```

```

57
58     Complex operator-(const Complex &B)
59     {
60         /// this -> A instance
61         Complex temp;
62         temp.real = this->real - B.real;
63         temp.imag = this->imag - B.imag;
64         return temp;
65     }

```

```

73 void print()
74 {
75     printf("[%d + i%d]\n", this->real, this->imag);
76 }
77 };

```



```
Supra-LLD > Codes > cpp > G: CompileTimePolymorphism.cpp > main()
70     }
71 };
72
73 int main()
74 {
75     Complex A(2, 5);
76     A.print();
77     Complex B(3, 3);
78     B.print();
79
80     Complex C = A + B;
81     C.print();
82
83     Complex D = A - B;
84     D.print();
85 }
```

cd "/Users/lovebabbar/Desktop/Lakshay/Supra-LLD/Codes/cpp/output"  
./"CompileTimePolymorphism"  
lovebabbar@Loves-MacBook-Pro Lakshay % cd "/Users/lovebabbar/Desktop/Lakshay/Supra-LLD/Codes/cpp/output"  
lovebabbar@Loves-MacBook-Pro output % ./"CompileTimePolymorphism"  
[2 + i5]  
[3 + i3]  
[5 + i8]  
lovebabbar@Loves-MacBook-Pro output % cd "/Users/lovebabbar/Desktop/Lakshay/Supra-LLD/Codes/cpp/output"  
lovebabbar@Loves-MacBook-Pro output % ./"CompileTimePolymorphism"  
[2 + i5]  
[3 + i3]  
[5 + i8]  
lovebabbar@Loves-MacBook-Pro output %

---



```

30 class Complex
31 {
32 public:
33     int real;
34     int imag;
35
36     Complex()
37     {
38         real = imag = -1;
39     }
40
41     Complex(int r, int i) : real(r), imag(i){};
42
43     // syntax
44     // Ret_type operator <op> (args){
45     //     // mlkdmk
46     //     return <>
47     // }
48
49     Complex operator+(const Complex &B)
50     {
51         /// this -> A instance
52         Complex temp;
53         temp.real = this->real + B.real;
54         temp.imag = this->imag + B.imag;
55         return temp;
56     }
57
58     Complex operator-(const Complex &B)
59     {
60         /// this -> A instance
61         Complex temp;
62         temp.real = this->real - B.real;
63         temp.imag = this->imag - B.imag;
64         return temp;
65     }
66
67     bool operator==(const Complex &B)
68     {
69         /// this -> A instance
70         return (this->real == B.real) && (this->imag == B.imag);
71     }
72
73     void print()
74     {
75         printf("[%d + i%d]\n", this->real, this->imag);
76     }
77 };
78

```

```
79 int main()
80 {
81     Complex A(2, 5);
82     A.print();
83     Complex B(3, 3);
84     B.print();
85
86     bool a = A == B;
87     cout << a << endl;
88     // Complex C = A + B;
89     // C.print();
90
91     // Complex D = A - B;
92     // D.print();
93 }
```

```
lovebabbar@Loves-MacBook-Pro output % cd "/Users/lovebabbar/Desktop/Lakshay/Supra-LLD/Codes/cpp/output"
./"CompileTimePolymorphism"
lovebabbar@Loves-MacBook-Pro output % ./"CompileTimePolymorphism"
[2 + 15]
[3 + 13]
[5 + 18]
[-1 + 12]
lovebabbar@Loves-MacBook-Pro output % cd "/Users/lovebabbar/Desktop/Lakshay/Supra-LLD/Codes/cpp/output"
./"CompileTimePolymorphism"
lovebabbar@Loves-MacBook-Pro output % ./"CompileTimePolymorphism"
[2 + 15]
[3 + 13]
1
lovebabbar@Loves-MacBook-Pro output % cd "/Users/lovebabbar/Desktop/Lakshay/Supra-LLD/Codes/cpp/output"
./"CompileTimePolymorphism"
lovebabbar@Loves-MacBook-Pro output % ./"CompileTimePolymorphism"
[2 + 15]
[3 + 13]
```

# Operator Overloading

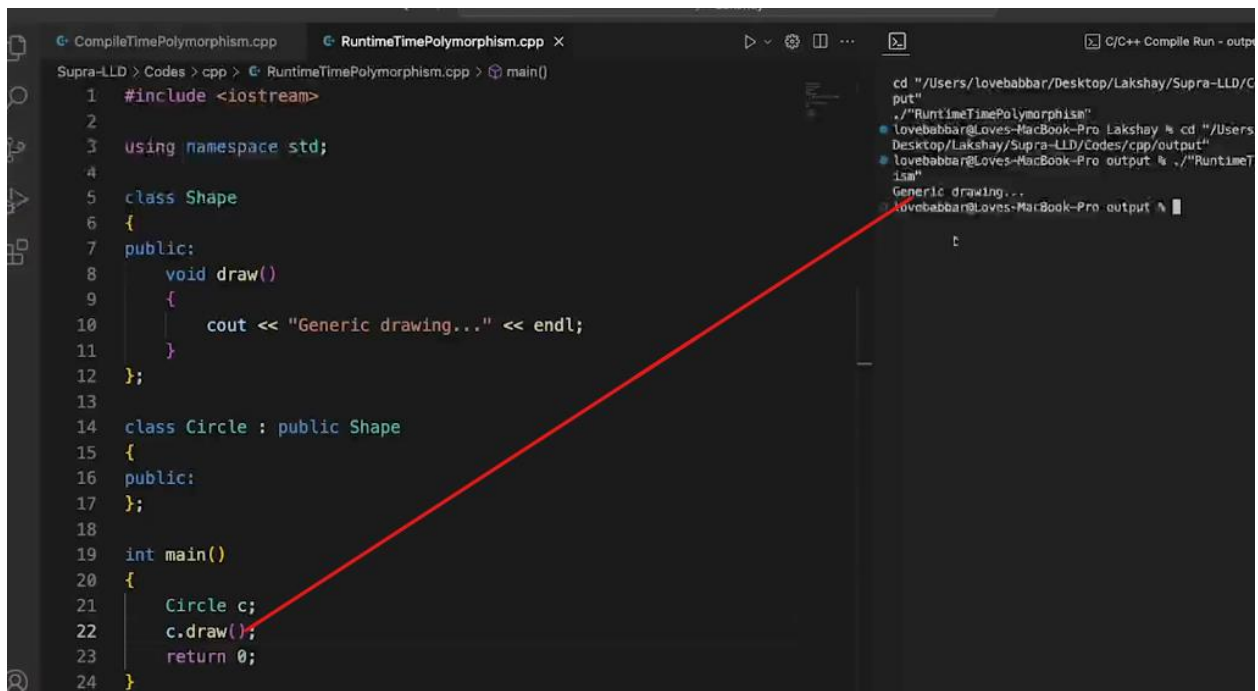
1. Below are the operators which can be overloaded in C++.

1. Arithmetic operators: '+', '-', '\*', '/', '%'
2. Relational operators: '==', '!=', '<', '>', '<=', '>='
3. Logical operators: '&&', '||', '!'
4. Assignment operators: '=', '+=', '-=', '\*=', '/=', '%=', '<<=', '>>=', '&=', '^=', '|='
5. Increment and decrement operators: '++', '--'
6. Subscript operator: '[]'
7. Function call operator: '()'
8. Member access operators: '->', '->\*'
9. Allocation and deallocation: 'new', 'new[]', 'delete', 'delete[]'
10. Bitwise operators: '&', '|', '^', '~', '<<', '>>'
11. Other: ',', '->\*', '()', '[]'

# Runtime Polymorphism

1. **Function Overriding** - makes function polymorphic.
2. Early vs Late binding.
3. Virtual Keyword - Way to achieve polymorphism by deferring binding decision to runtime.
4. Override keyword - Helps to make the intention clear and allows the compiler to enforce overriding rules, making your code safer and easier to understand.
5. Upcasting / Down-casting.

```
class Shape {
public:
    virtual void draw() {
        cout << "Drawing a generic shape" << endl;
    }
};
class Circle : public Shape {
public:
    void draw() override {
        cout << "Drawing a circle" << endl;
    }
};
class Rectangle : public Shape {
public:
    void draw() override {
        cout << "Drawing a rectangle" << endl;
    }
};
```



```
Supra-LLD > Codes > cpp > RuntimeTimePolymorphism.cpp > main()
1  #include <iostream>
2
3  using namespace std;
4
5  class Shape
6  {
7  public:
8      void draw()
9      {
10         cout << "Generic drawing..." << endl;
11     }
12 };
13
14 class Circle : public Shape
15 {
16 public:
17 };
18
19 int main()
20 {
21     Circle c;
22     c.draw();
23     return 0;
24 }
```

```
cd "/Users/lovebabbar/Desktop/Lakshay/Supra-LLD/Codes/cpp/output"
./"RuntimeTimePolymorphism"
lovebabbar@Loves-MacBook-Pro Lakshay % cd "/Users/Desktop/Lakshay/Supra-LLD/Codes/cpp/output"
lovebabbar@Loves-MacBook-Pro output % ./"RuntimeTimePolymorphism"
Generic drawing...
lovebabbar@Loves-MacBook-Pro output %
```

```
Supra-LLD > Codes > cpp > RuntimeTimePolymorphism.cpp > Circle > draw()

4
5 class Shape
6 {
7 public:
8     void draw()
9     {
10         cout << "Generic drawing..." << endl;
11     }
12 };
13
14 class Circle : public Shape
15 {
16 public:
17     void draw()
18     {
19         cout << "Circle drawing..." << endl;
20     }
21 };
22
23 int main()
24 {
25     Circle c;
26     c.draw();
27     return 0;
28 }
29
```

cd "/Users/lovebabbar/Desktop/Lakshay/Supra-LLD/Code  
put"  
./"RuntimeTimePolymorphism"  
lovebabbar@Loves-MacBook-Pro Lakshay % cd "/Users/lo  
Desktop/Lakshay/Supra-LLD/Codes/cpp/output"  
lovebabbar@Loves-MacBook-Pro output % ./"RuntimeTime  
ism"  
Circle drawing...  
lovebabbar@Loves-MacBook-Pro output %

Compiled successfully!

---

Supra-LLD > Codes > cpp > RuntimeTimePolymorphism.cpp > Circle > draw()

```
1  #include <iostream>
2
3  using namespace std;
4
5  class Shape
6  {
7  public:
8      void draw()
9      {
10         cout << "Generic drawing..." << endl;
11     }
12 };
13
14 class Circle : public Shape
15 {
16 public:
17     void draw()
18     {
19         cout << "Circle drawing..." << endl;
20     }
21 };
22
```

```
23 class Rectangle : public Shape
24 {
25 public:
26     void draw()
27     {
28         cout << "Rectangle drawing..." << endl;
29     }
30 };
31
32 void ShapeDrawing(Shape *s)
33 {
34     s->draw();
35 }
36
```

```
Supra-LLD > Codes > cpp > RuntimeTimePolymorphism.cpp > main()
27 {
28     cout << "Rectangle drawing.." << endl;
29 }
30 };
31
32 void ShapeDrawing(Circle *s)
33 {
34     s->draw();
35 }
36
37 int main()
38 {
39     Circle c;
40     Rectangle r;
41     ShapeDrawing(&c);
42     return 0;
43 }
44
```

cd "/Users/lovebabbar/Desktop/Lakshay/Supra-LLD/put"  
./"RuntimeTimePolymorphism"  
lovebabbar@Loves-MacBook-Pro Lakshay % cd "/Users/Desktop/Lakshay/Supra-LLD/Codes/cpp/output"  
lovebabbar@Loves-MacBook-Pro output % ./"Runtimeism"  
Circle drawing...  
lovebabbar@Loves-MacBook-Pro output %

---

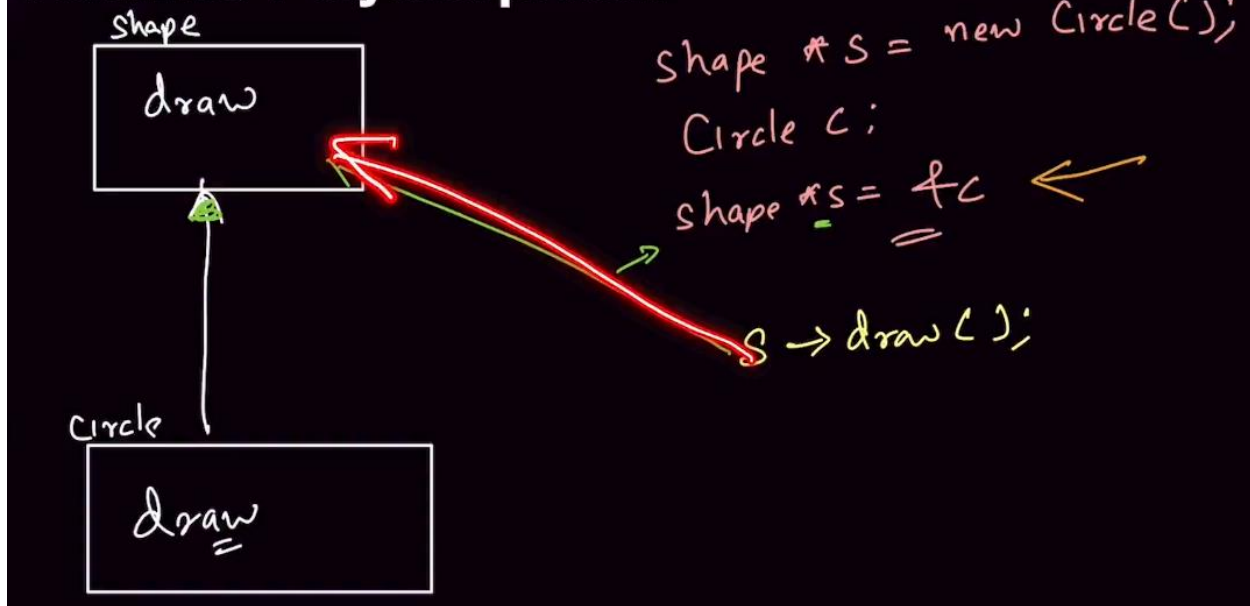
```
Supra-LLD > Codes > cpp > RuntimeTimePolymorphism.cpp > ShapeDrawing(Shape *)
24 {
25     public:
26     void draw()
27     {
28         cout << "Rectangle drawing.." << endl;
29     }
30 };
31
32 void ShapeDrawing(Shape *s)
33 {
34     s->draw();
35 }
36
37 int main()
38 {
39     Circle c;
40     Rectangle r;
41
42     ShapeDrawing(&c);
43     ShapeDrawing(&r);
44     return 0;
45 }
46
```

cd "/Users/lovebabbar/Desktop/Lakshay/Supra-LLD/Codes/cpp/put"  
./"RuntimeTimePolymorphism"  
lovebabbar@Loves-MacBook-Pro Lakshay % cd "/Users/lovebabbar/Desktop/Lakshay/Supra-LLD/Codes/cpp/output"  
lovebabbar@Loves-MacBook-Pro output % ./"RuntimeTimePolymorphism"  
Generic drawing...  
Generic drawing...  
lovebabbar@Loves-MacBook-Pro output %

Iss code mai problem hai kyuki hum ne child ka address send kiya hai shape class ke pointer ko.

---

# Runtime Polymorphism



Iss mai shape ka draw()

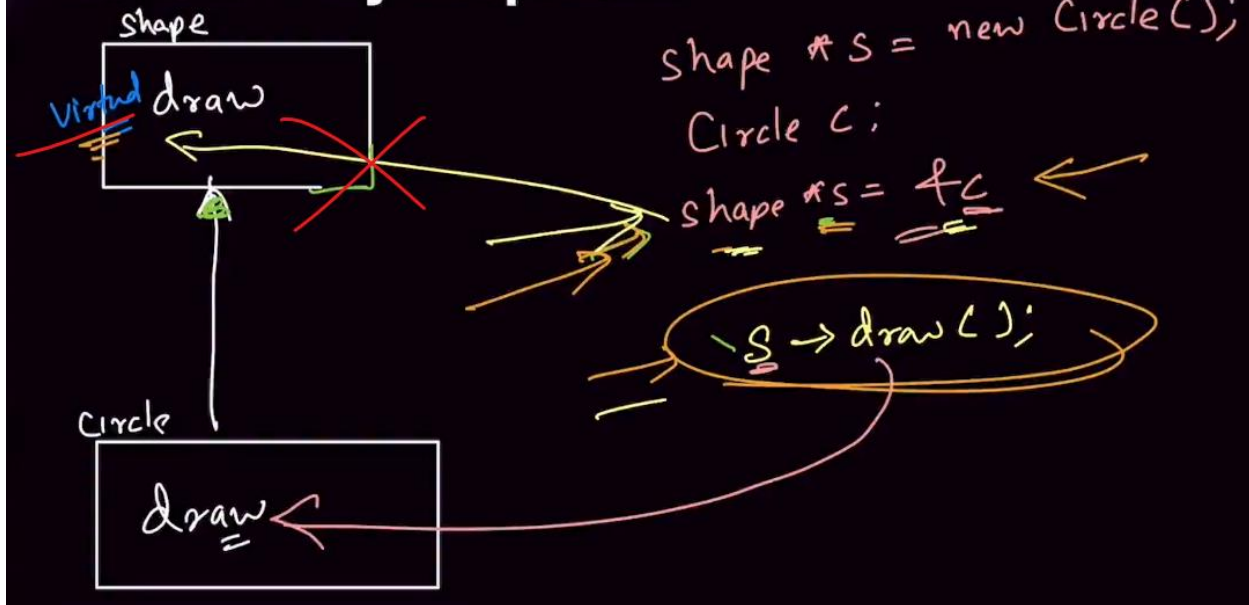
Compiler ne dekha ki shape class ka object bana hai toh shape class ke draw() ko call kar dega iss ko Early binding kahte hai (compile time pe binding ho gayi)

---

```
Circle c;  
Shape *s=&c;  
upcasting : parent class ka pointer child class ko hold kar raha hai
```



# Runtime Polymorphism



Runtime pe decision hoga ki kis method ko call karna ? shape class ke method virtual ho gaya hai aur shape class ka object child class ke address ko hold kiya ,to hisss mai child class ka method call hoga aur isko hum LATE binding ya runtime polymorphism bhi kahte hai.

**Late binding** means runtime pe binding hogi.

**Early binding** means compile time pe binding hogi.

---

```
Supra-LLD > Codes > cpp > RuntimeTimePolymorphism.cpp > Shape > draw()
1  #include <iostream>
2
3  using namespace std;
4
5  class Shape
6  {
7  public:
8      virtual void draw()
9      {
10         cout << "Generic drawing..." << endl;
11     }
12 };
13
14 class Circle : public Shape
15 {
16 public:
17     void draw()
18     {
19         cout << "Circle drawing..." << endl;
20     }
21 };
22
23 class Rectangle : public Shape
24 {
25 public:
26     void draw()
27     {
```

```
CompileTimePolymorphism.cpp  RuntimeTimePolymorphism.cpp x
Supra-LLD > Codes > cpp > RuntimeTimePolymorphism.cpp > ShapeDrawing(Shape *)
33 {
34 public:
35     void draw()
36     {
37         cout << "Triangle drawing..." << endl;
38     }
39 };
40
41 void ShapeDrawing(Shape *s)
42 {
43     s->draw();
44 }
45
46 int main()
47 {
48     Circle c;
49     Rectangle r;
50
51     ShapeDrawing(&c);
52     ShapeDrawing(&r);
53
54     Triangle *t = new Triangle();
55     ShapeDrawing(t);
56     return 0;
57 }
58

cd "/Users/lovebabbar/Desktop/Lakshay/Supra-LLD/Code
put"
./"RuntimeTimePolymorphism"
lovebabbar@Loves-MacBook-Pro Lakshay % cd "/Users/lo
Desktop/Lakshay/Supra-LLD/Codes/cpp/output"
lovebabbar@Loves-MacBook-Pro output % ./"RuntimeTime
ism"
circle drawing...
Rectangle drawing...
Triangle drawing...
lovebabbar@Loves-MacBook-Pro output %

Compiled successfully!
```

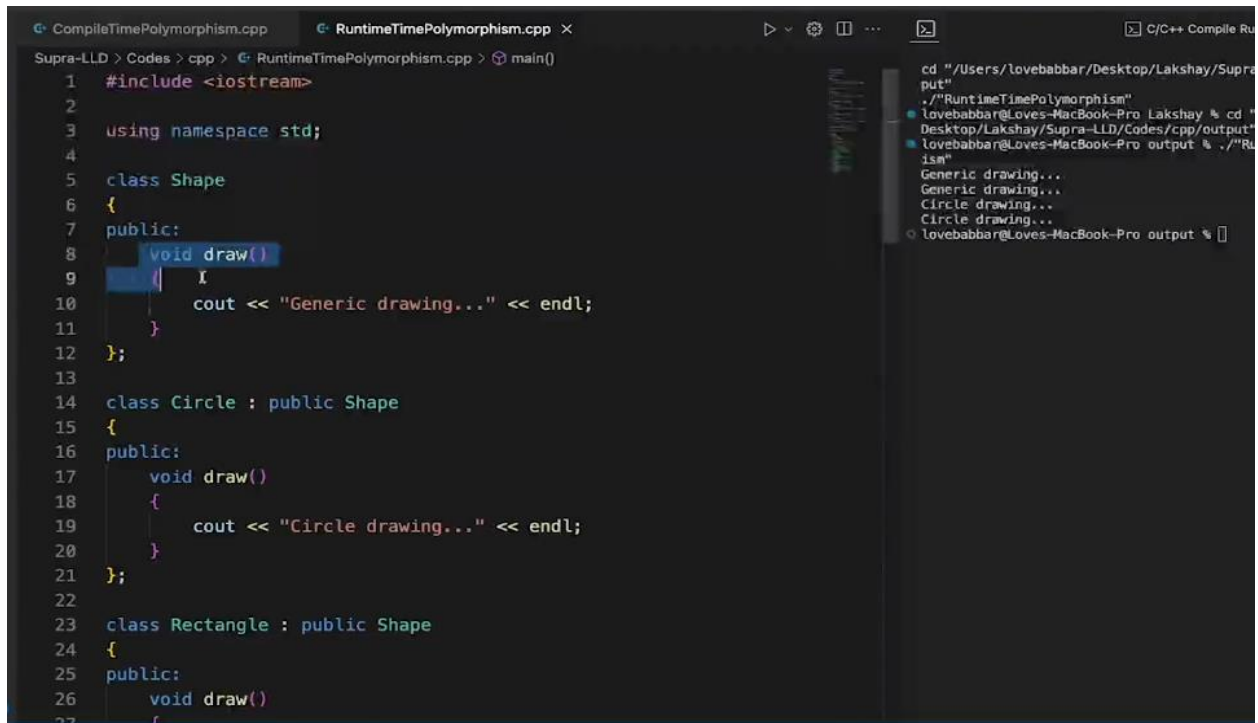
---

Without virtual method bana hai shape class ke method mai

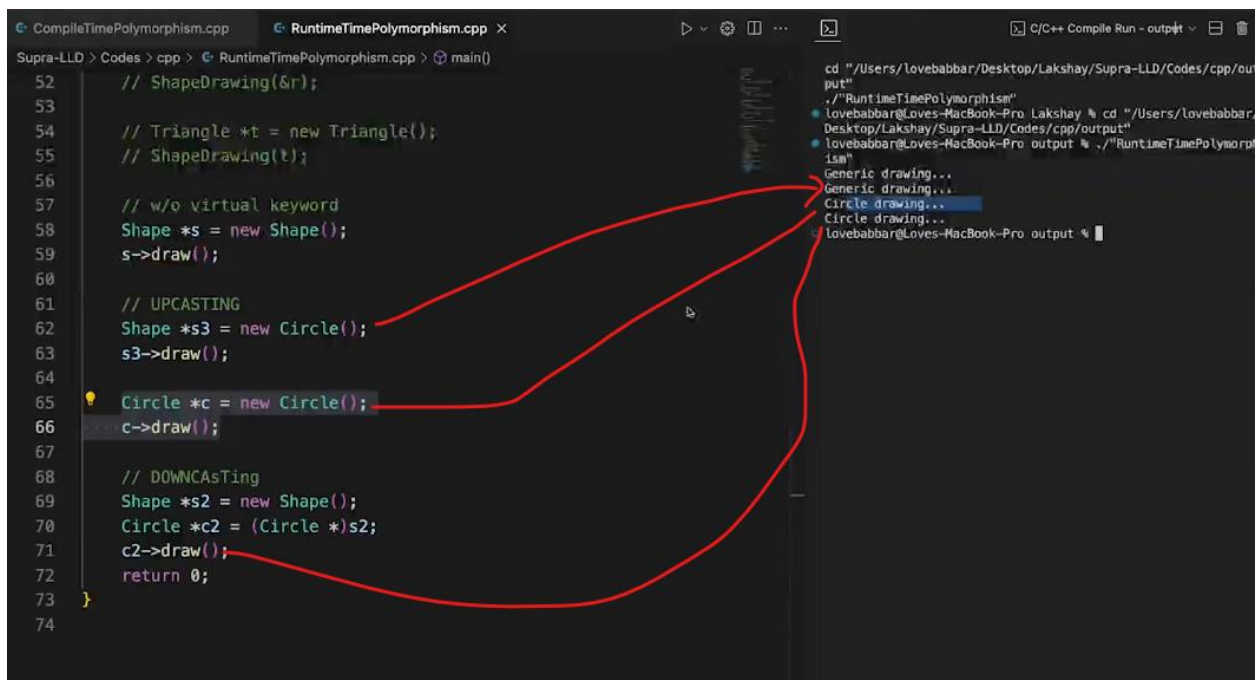
```
Shape *s = new shape ()
```

```
s.draw()
```

shape class ka draw() call hoga.



```
1 #include <iostream>
2
3 using namespace std;
4
5 class Shape
6 {
7 public:
8     void draw()
9     {
10         cout << "Generic drawing..." << endl;
11     }
12 };
13
14 class Circle : public Shape
15 {
16 public:
17     void draw()
18     {
19         cout << "Circle drawing..." << endl;
20     }
21 };
22
23 class Rectangle : public Shape
24 {
25 public:
26     void draw()
27 {
```



```
52 // ShapeDrawing(&r);
53
54 // Triangle *t = new Triangle();
55 // ShapeDrawing(t);
56
57 // w/o virtual keyword
58 Shape *s = new Shape();
59 s->draw();
60
61 // UPCASTING
62 Shape *s3 = new Circle();
63 s3->draw();
64
65 Circle *c = new Circle();
66 c->draw();
67
68 // DOWNCASTING
69 Shape *s2 = new Shape();
70 Circle *c2 = (Circle *)s2;
71 c2->draw();
72 return 0;
73 }
74
```

# Runtime Polymorphism <sup>important</sup>

W/o virtual

↳ Left me jo likha hai

↳ jiske obj me actual obj store

Class A = new class B;

W/ virtual

↳ actual jo hona chahiye  
jo actual obj bana hai

With virtual keyword

```
LLD > Codes > cpp > RuntimeTimePolymorphism.cpp > main()

// Rectangle r;

// ShapeDrawing(&c);
// ShapeDrawing(&r);

// Triangle *t = new Triangle();
// ShapeDrawing(t);

// virtual keyword
Shape *s = new Shape();
s->draw();

// UPCASTING
Shape *s3 = new Circle();
s3->draw();

Circle *c = new Circle();
c->draw();

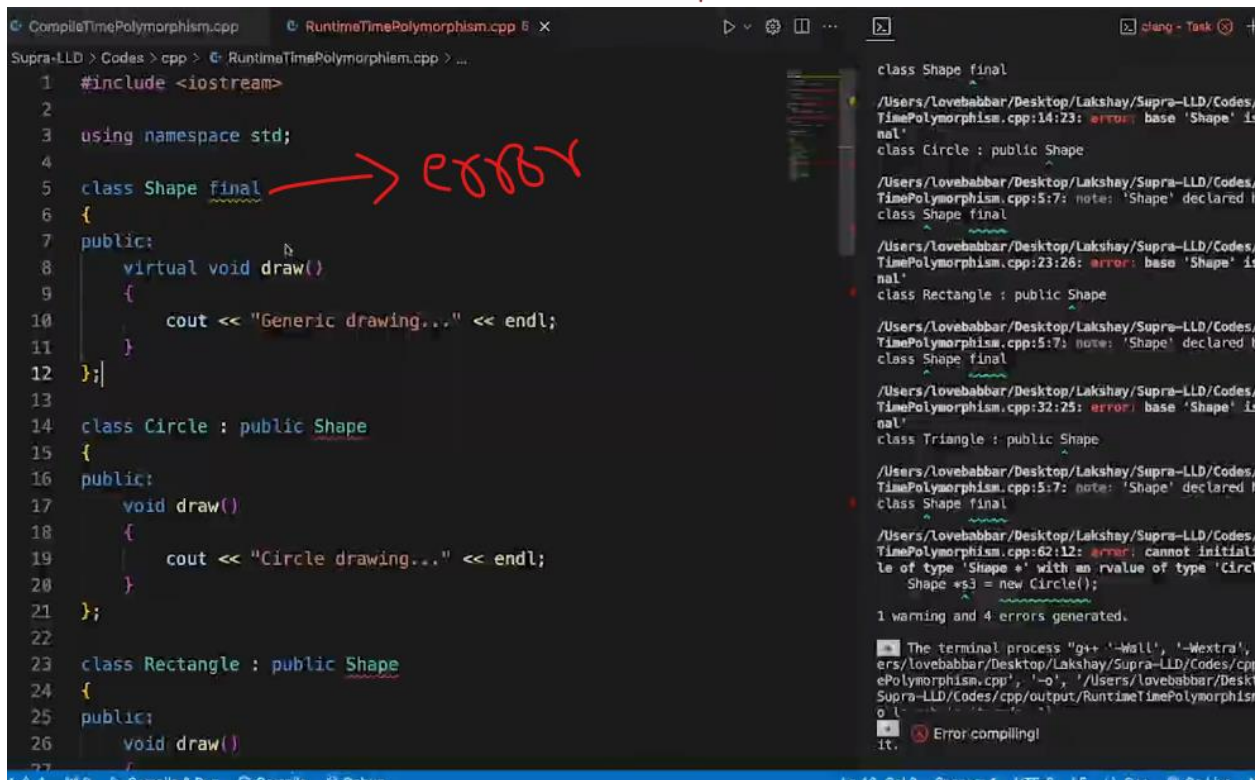
// DOWNCASTING
Shape *s2 = new Shape();
Circle *c2 = (Circle *)s2;
c2->draw();
return 0;
}
```

cd "/Users/lovebabbar/Desktop/Lakshay/Supra-LLD/output"  
./"RuntimeTimePolymorphism"  
lovebabbar@Loves-MacBook-Pro Lakshay % cd "/Users/Desktop/Lakshay/Supra-LLD/Codes/cpp/output"  
lovebabbar@Loves-MacBook-Pro output % ./"RuntimeTimePolymorphism"  
Generic drawing...  
Generic drawing...  
Circle drawing...  
Circle drawing...  
lovebabbar@Loves-MacBook-Pro output % cd "/Users/Desktop/Lakshay/Supra-LLD/Codes/cpp/output"  
lovebabbar@Loves-MacBook-Pro output % ./"RuntimeTimePolymorphism"  
Generic drawing...  
Circle drawing...  
Circle drawing...  
Circle drawing...  
Generic drawing...  
lovebabbar@Loves-MacBook-Pro output %

shape\* s=new shape();  
Circle \* C2=(Circle \*)s;  
downcasting : child ka pointer parent ke object ko hold kiya hai;

# Final Keyword

1. In C++, the final specifier is used in two main contexts: with classes and with virtual member functions.
2. **Prevents Class Inheritance**: When you declare a class as final, it means that no other class can inherit from it.
3. **Preventing Virtual Function Overriding**: The final specifier can also be used with virtual functions to prevent them from being overridden in derived classes.



```
1 #include <iostream>
2
3 using namespace std;
4
5 class Shape final
6 {
7 public:
8     virtual void draw()
9     {
10         cout << "Generic drawing..." << endl;
11     }
12 };
13
14 class Circle : public Shape
15 {
16 public:
17     void draw()
18     {
19         cout << "Circle drawing..." << endl;
20     }
21 };
22
23 class Rectangle : public Shape
24 {
25 public:
26     void draw()
```

The screenshot shows a C++ IDE with a code editor on the left and a compiler error log on the right. The code defines a base class `Shape` with a `draw()` method, and two derived classes, `Circle` and `Rectangle`, both inheriting from `Shape`. The `Shape` class is marked as `final`. A red arrow points to the `final` keyword with the word "error" written in red. The error log on the right shows multiple compilation errors, including "error: base 'Shape' is final" and "error: cannot initialize of type 'Shape\*' with an rvalue of type 'Circle'", indicating that the derived classes cannot inherit from the final base class.



Supra-LLD > Codes > cpp > RuntimeTimePolymorphism.cpp > Circle > draw()

```

4
5 class Shape
6 {
7 public:
8     virtual void draw() final
9     {
10         cout << "Generic drawing..." << endl;
11     }
12 };
13     declaration of 'draw' overrides a 'final'
14     function gcc
15 class Circle
16 {
17 public:
18     inline virtual void Circle::draw()
19     void draw()
20     {
21         cout << "Circle drawing..." << endl;
22     }
23 };
24
25 class Rectangle : public Shape
26 {
27 public:
28     void draw()
29     {
30         cout << "Rectangle drawing..." << endl;
31     }
32 };

```

```
TimePolymorphism.cpp:8:25: warning: 'final' keyword is
1 extension [-Wc++11-extensions]
    virtual void draw() final
```

```
/Users/lovebabbar/Desktop/Lakshay/Supra-L1D/Codes/cpp/Re
TimePolymorphism.cpp:17:10: error: declaration of 'draw
rides a 'final' function
void draw()
```

```
/Users/Toyeabbar/Desktop/Lakshay/Supra-LLD/Codes/cpp/RunTimePolymorphism.cpp:8:18: notes overridden virtual fun
is here
```

```
/Users/Lovebabbar/Desktop/Lakshay/Supra-LLD/Codes/cpp/Re
TimePolymorphism.cpp:26:10: error: declaration of 'draw'
rides a 'final' function
    void draw()
```

```
/Users/lovebabbar/Desktop/Lakshay/Supra-LLD/Codes/cpp/RuntimePolymorphism.cpp:8:18: note: overridden virtual function is here
    virtual void draw() final
```

```
/Users/lovebabbar/Desktop/Lakshay/Supra-LIB/Codes/cpp/RT
TimePolymorphism.cpp:35:10: error: declaration of 'draw'
rides a 'final' function
    void draw()
```

```
/Users/lovebabbar/Desktop/Lakshay/Supra-115/Codes/cpp/8
TimePolymorphism.cpp:8:18: note: overridden virtual fun
is here
    virtual void draw() final
```

1 warning and 3 errors generated.

```
* The terminal process "g++ -Wall, -Wextra, -g3  
ers/lovebabbar/Desktop/Lakshay/Supra-LLD/Codes/cpp/Run  
ePolymorphism.cpp", "-o", "/Users/lovebabbar/Desktop/La  
Supra-LLD/Codes/cpp/output/RuntimeTimePolymorphism" fa
```

 Error compiling!

## 8. Abstraction In C++

main.cpp C bird.h 2 x  
E: > LLD\_Notes > 2.Object Oriented Systems(OOPS) > 8.Abstraction in C++ [Codes] 2 > Abstraction in C++ [Codes] 2 > C bird.h > ...

```
1  #ifndef BIRD_H
2  #define BIRD_H
3  #include <bits/stdc++.h>
4  class Bird
5  {
6  public:
7      // this is interface
8      virtual void eat() = 0;
9      virtual void fly() = 0;
10     // classes that inherits this class has to implement pure virtual functions
11 };
12 class Sparrow : public Bird
13 {
14 public:
15     void eat()
16     {
17         std::cout << "Sparrow is eating\n";
18     }
19     void fly()
20     {
21         std::cout << "Sparrow is flying \n";
22     }
23 };
24 class Eagle : public Bird
25 {
26 public:
27     void eat()
28     {
29         std::cout << "Eagle is eating\n";
30     }
31     void fly()
32     {
33         std::cout << "Eagle is flying \n";
34     }
35 };
36 class Pigeon : public Bird
37 {
38 public:
39     void eat()
40     {
41         std::cout << "Pigeon is eating\n";
42     }
43     void fly()
44     {
45         std::cout << "Pigeon is flying \n";
46     }
47 };
48 #endif
```



```
main.cpp 2 x  C bird.h 2
E: > LLD_Notes > 2.Objected Oriented Systems(OOPS) > 8.Abstraction in C++ [Codes] 2 > Abstraction in C++ [Codes] 2 > main.cpp > ...
1  #include <bits/stdc++.h>
2  #include "bird.h"
3  using namespace std;
4  void birdDoesSomething(Bird *&bird)
5  {
6      bird->eat();
7      bird->fly();
8  }
9  int main()
10 {
11     // Bird *bird = new Sparrow();
12     // Bird *bird = new Eagle();
13     Bird *bird = new Pigeon();
14     birdDoesSomething(bird);
15     // interface same h but functionality alag ho skti h ya changes ho skte h humme code mai koi change nhi krna
16     return 0;
17 }
18
```

Abstraction ya Inteface class ka OBJECT nahi banata hai .

---